

DD2434 Machine Learning, Advanced Course

Assignment 1

Jiang, Sifan
sifanj@kth.se

January 5, 2020

Contents

1	Knowing the Rules	2
2	Dependencies in a Directed Graphical Model	2
3	Likelihood of a tree GM only for E level	3
4	Simple VI	3
5	Mixture of trees with observable variables	8
6	Super epicentra - EM	12
7	Super epicentra - VI	17
8	Sampling from a tree GM	20
9	Failing components VI	20

Listings

1	Simple VI.	4
2	EM algorithm for mixture of trees with observable variables.	9
3	EM algorithm for super epicentra.	15

1 Knowing the Rules

Question 2.1.1: *It is mandatory to read the above text. Have you read it?*

Yes.

Question 2.1.2: *List all your collaborations concerning the problem formulations in this assignment.*

Question 2.1.3: *Have you discussed solutions with anybody?*

No.

2 Dependencies in a Directed Graphical Model

Question 2.2.4: *In the graphical model of Figure 5, is $\mu_k \perp \tau_k$ (not conditioned by anything)?*

Yes, $\mu_k \perp \tau_k$ is true.

$$p(\mu_k, \tau_k) = p(\mu_k)p(\tau_k)p(X^1, \dots, X^N \mid \mu_k, \tau_k).$$

Since none of the variables are observed, after “marginalizing both sides of” the above equation over X^1, \dots, X^N we obtain [1]:

$$p(\mu_k, \tau_k) = p(\mu_k)p(\tau_k).$$

So, $\mu_k \perp \tau_k$.

Question 2.2.5: *In the graphical model of Figure 5, is $\mu_k \perp \tau_k \mid X^1, \dots, X^N$?*

No, $\mu_k \perp \tau_k \mid X^1, \dots, X^N$ is false.

$$\begin{aligned} p(\mu_k, \tau_k \mid X^1, \dots, X^N) &= \frac{p(\mu_k, \tau_k, X^1, \dots, X^N)}{p(X^1, \dots, X^N)} \\ &= \frac{p(\mu_k)p(\tau_k)p(X^1, \dots, X^N \mid \mu_k, \tau_k)}{p(X^1, \dots, X^N)}, \end{aligned}$$

which isn't factorized into the product $p(\mu_k)p(\tau_k)$, thus $\mu_k \perp \tau_k \mid X^1, \dots, X^N$ is false.

Question 2.2.6: *In the graphical model of Figure 8, is $\mu \perp b'$ (not conditioned by anything)?*

No, $\mu \perp b'$ is false.

$$p(\mu, b', \tau) = p(b')p(\tau \mid b')p(\mu \mid \tau).$$

By marginalizing over τ , we can obtain

$$p(\mu, b') = p(b') \sum_{\tau} p(\tau \mid b')p(\mu \mid \tau) = p(\mu \mid b')p(b'),$$

which isn't factorized into the product $p(\mu)p(b')$, thus $\mu \perp b'$ is false.

Question 2.2.7: In the graphical model of Figure 8, is $\mu \perp b' \mid X^1, \dots, X^N$?

No.

Question 2.2.8: In the graphical model of Figure 8, is $X^n \perp S^n$ (not conditioned by anything)?

No.

Question 2.2.9: In the graphical model of Figure 8, is $X^n \perp S^n \mid \mu_k, \tau_k$?

No.

3 Likelihood of a tree GM only for E level

4 Simple VI

Question 2.4.12: Implement the VI algorithm for the variational distribution in Equation (10.24) in Bishop.

From Bishop [1], the variational distribution is given by

$$\begin{aligned} q(\mu, \tau) &= q_\mu(\mu)q_\tau(\tau) \\ q_\mu(\mu) &= \mathcal{N}(\mu \mid \mu_N, \lambda_N^{-1}) \\ q_\tau(\tau) &= \text{Gam}(\tau \mid a_N, b_N), \end{aligned}$$

where

$$\begin{aligned} \mu_N &= \frac{\lambda_0 \mu_0 + N \bar{x}}{\lambda_0 + N} \\ \lambda_N &= (\lambda_0 + N) \mathbb{E}[\tau] \\ a_N &= a_0 + \frac{N}{2} \\ b_N &= b_0 + \frac{1}{2} \mathbb{E}_\mu \left[\sum_{n=1}^N (x_n - \mu)^2 + \lambda_0 (\mu - \mu_0)^2 \right] \\ &= b_0 - \left(\sum_{n=1}^N x_n + \lambda_0 \mu_0 \right) \mathbb{E}_\mu[\mu] + \frac{1}{2} \left(\sum_{n=1}^N x_n^2 + \lambda_0 \mu_0^2 + (\lambda_0 + N) \mathbb{E}_\mu[\mu^2] \right). \end{aligned}$$

Where

$$\begin{aligned} \mathbb{E}_\mu[\mu] &= \mu_N \\ \mathbb{E}_\mu[\mu^2] &= \frac{1}{\lambda_N} + \mu_N^2 \\ \mathbb{E}_\tau[\tau] &= \frac{a_N}{b_N}. \end{aligned}$$

Due to non-informative condition, we set $\mu_0 = \lambda_0 = a_0 = b_0 = 0$. Then the VI algorithm would be an iterative algorithm that start with a set of initial values of μ_N , λ_N , a_N , and b_N . Then the values of μ_N , λ_N , a_N , and b_N should be updated to obtain a converging result of approximated posterior distribution. The algorithm is shown in the following code:

```

1 from math import exp, pi, sqrt
  from scipy.special import gamma

  import matplotlib.pyplot as plt
  import numpy as np

6 np.random.seed(1000)
  MAXITER = 1000
  THRES = 0.001

11 def _q(muN, lambdaN, aN, bN, mu, tau):
    q_mu = sqrt(lambdaN/(2*pi)) * np.exp(-0.5 * np.dot(lambdaN, np.transpose((mu-muN)
    )**2)))
    q_tau = (1.0/gamma(aN)) * bN**aN * tau**(aN-1) * np.exp(-bN*tau)
    q = q_tau * q_mu
    return q

16 def _update(D, N, mu0, lambda0, a0, b0, muN, lambdaN, aN, bN):
    E_mu = muN
    E_mu2 = 1.0 / lambdaN + muN**2
    E_tau = aN / bN
    lambdaN = (lambda0 + N) * E_tau
    bN = b0 - (sum(D) + lambda0*mu0)*E_mu \
    + 0.5*(sum(D**2) + lambda0*mu0**2 + (lambda0+N)*E_mu2)
    return lambdaN, bN

26 def SimpleVI():
    # Generate data set.
    mu_D = 0.0
    sigma_D = 1.0
    N = 100
    D = np.random.normal(loc=mu_D, scale=sigma_D, size=N)

    # Initial values.
    x_bar = D.mean()
    mu0 = 0
    lambda0 = 0
    a0 = 0
    b0 = 0

    muN = (lambda0*mu0 + N*x_bar) / (lambda0 + N)
    lambdaN = 10
    aN = a0 + N / 2
    bN = 5

    lambdaOld = lambdaN
    bOld = bN

    for _ in range(MAXITER):
        lambdaN, bN = _update(D, N, mu0, lambda0, a0, b0, muN, lambdaN, aN, bN)
        if (abs(lambdaN - lambdaOld) < THRES) and (abs(bN - bOld) < THRES):
            break
        lambdaOld = lambdaN

```

56

```

        bOld = bN

    return muN, lambdaN, aN, bN

if __name__ == "__main__":
    muN, lambdaN, aN, bN = SimpleVI()

```

Listing 1: Simple VI.

Question 2.4.13: *What is the exact posterior?*

Since gamma distribution is a conjugate prior of the normal distribution, the joint distribution of the Normal-Gamma distribution is

$$p(\mu, \tau \mid \mu_0, \lambda_0, a_0, b_0) = \frac{b_0^{a_0} \sqrt{\lambda_0}}{\Gamma(a_0) \sqrt{2\pi}} \tau^{a_0 - \frac{1}{2}} \exp[-b_0 \tau] \exp\left[-\frac{\lambda \tau (\mu - \mu_0)^2}{2}\right]$$

$$p(\mu, \tau) \propto \tau^{a_0 - \frac{1}{2}} \exp[-b_0 \tau] \exp\left[-\frac{\lambda \tau (\mu - \mu_0)^2}{2}\right]$$

According to Bayesian theorem, we have

$$p(\mu, \tau \mid \mathcal{D}) \propto p(\mathcal{D} \mid \mu, \tau) p(\mu, \tau),$$

where $p(\mathcal{D} \mid \mu, \tau)$ is the likelihood of the data points which have such relationship:

$$\begin{aligned}
 p(\mathcal{D} \mid \mu, \tau) &= \prod_{i=1}^N p(x_i \mid \mu, \tau) \\
 &\propto \prod_{i=1}^N \tau^{1/2} \exp\left[-\frac{\tau}{2}(x_i - \mu)^2\right] \\
 &\propto \tau^{N/2} \exp\left[-\frac{\tau}{2} \sum_{i=1}^N (x_i - \mu)^2\right] \\
 &\propto \tau^{N/2} \exp\left[-\frac{\tau}{2} \sum_{i=1}^N (x_i - \bar{x} + \bar{x} - \mu)^2\right] \\
 &\propto \tau^{N/2} \exp\left[-\frac{\tau}{2} \sum_{i=1}^N ((x_i - \bar{x})^2 + (\bar{x} - \mu)^2)\right] \\
 &\propto \tau^{N/2} \exp\left[-\frac{\tau}{2} (Ns + N(\bar{x} - \mu)^2)\right],
 \end{aligned}$$

where $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$ and $s = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$. Thus the posterior can be written into [2]

$$\begin{aligned}
 p(\mu, \tau \mid \mathcal{D}) &\propto p(\mathcal{D} \mid \mu, \tau) p(\mu, \tau) \\
 &\propto \tau^{N/2} \exp\left[-\frac{\tau}{2} (Ns + N(\bar{x} - \mu)^2)\right] \tau^{a_0 - \frac{1}{2}} \exp[-b_0 \tau] \exp\left[-\frac{\lambda \tau (\mu - \mu_0)^2}{2}\right] \\
 &\propto \tau^{\frac{N}{2} + a_0 - \frac{1}{2}} \exp\left[-\tau \left(\frac{1}{2} Ns + b_0\right)\right] \exp\left[-\frac{\tau}{2} (\lambda_0 (\mu - \mu_0)^2 + N(\bar{x} - \mu)^2)\right].
 \end{aligned}$$

The exponent (except the $-\frac{\tau}{2}$) in the last exponential term can be simplified into

$$\begin{aligned}
\lambda_0(\mu - \mu_0)^2 + N(\bar{x} - \mu)^2 &= \lambda_0\mu^2 - 2\lambda_0\mu\mu_0 + \lambda_0\mu_0^2 + N\mu^2 - 2N\bar{x}\mu + N\bar{x}^2 \\
&= (\lambda_0 + N)\mu^2 - 2(\lambda_0\mu_0 + N\bar{x})\mu + \lambda_0\mu_0^2 + N\bar{x}^2 \\
&= (\lambda_0 + N) \left(\mu^2 - 2\frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N}\mu \right) + \lambda_0\mu_0^2 + N\bar{x}^2 \\
&= (\lambda_0 + N) \left(\mu - \frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N} \right)^2 + \lambda_0\mu_0^2 + N\bar{x}^2 - \frac{(\lambda_0\mu_0 + N\bar{x})^2}{\lambda_0 + N} \\
&= (\lambda_0 + N) \left(\mu - \frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N} \right)^2 + \frac{\lambda_0N(\bar{x} - \mu_0)^2}{\lambda_0 + N}.
\end{aligned}$$

Insert the above equation into the posterior distribution, we have

$$\begin{aligned}
p(\mu, \tau \mid \mathcal{D}) &\propto \tau^{\frac{N}{2} + a_0 - \frac{1}{2}} \exp \left[-\tau \left(\frac{1}{2}Ns + b_0 \right) \right] \exp \left[-\frac{\tau}{2} \left((\lambda_0 + N) \left(\mu - \frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N} \right)^2 + \frac{\lambda_0N(\bar{x} - \mu_0)^2}{\lambda_0 + N} \right) \right] \\
&\propto \tau^{\frac{N}{2} + a_0 - \frac{1}{2}} \exp \left[-\tau \left(\frac{1}{2}Ns + b_0 + \frac{\lambda_0N(\bar{x} - \mu_0)^2}{2(\lambda_0 + N)} \right) \right] \exp \left[-\frac{\tau}{2}(\lambda_0 + N) \left(\mu - \frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N} \right)^2 \right],
\end{aligned}$$

which is in the form of a Normal-Gamma distribution, thus the exact posterior is

$$\begin{aligned}
p(\mu, \tau \mid \mathcal{D}) &= \text{NormalGamma}(\mu', \lambda', a', b') \\
\mu' &= \frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N} \\
\lambda' &= \lambda_0 + N \\
a' &= a_0 + \frac{N}{2} \\
b' &= b_0 + \frac{1}{2} \left(Ns + \frac{\lambda_0N(\bar{x} - \mu_0)^2}{\lambda_0 + N} \right).
\end{aligned}$$

Question 2.4.14: Compare the inferred variational distribution with the exact posterior. Run the inference on data points drawn from iid Gaussians. Do this for three interesting cases and visualize the results. Describe the differences.

1. In this case, the number of data set is $N = 10$ and the initial values are $b_N = 5$ and $\lambda_N = 5$. The result is shown in fig 1, where the true posterior distribution is shown by the colorful contour lines, while the approximated distribution given by the VI algorithm is shown in blue contour lines and red contour lines when reached convergence.

Although we are using zero for parameters in the prior distributions, so that $\mu_0 = \lambda_0 = a_0 = b_0 = 0$, which implies that we have no information about the distribution, the convergence is really quick with VI algorithm.

And not only from this case, but also from the following cases, we can see that the mean of the μ is always correct, which is true since we get value μ_N at the beginning and don't update it during the iterations.

2. In this case, the number of data set is $N = 10$ and the initial values are $b_N = 0.1$ and $\lambda_N = 0.1$. The result is illustrated in fig 2.

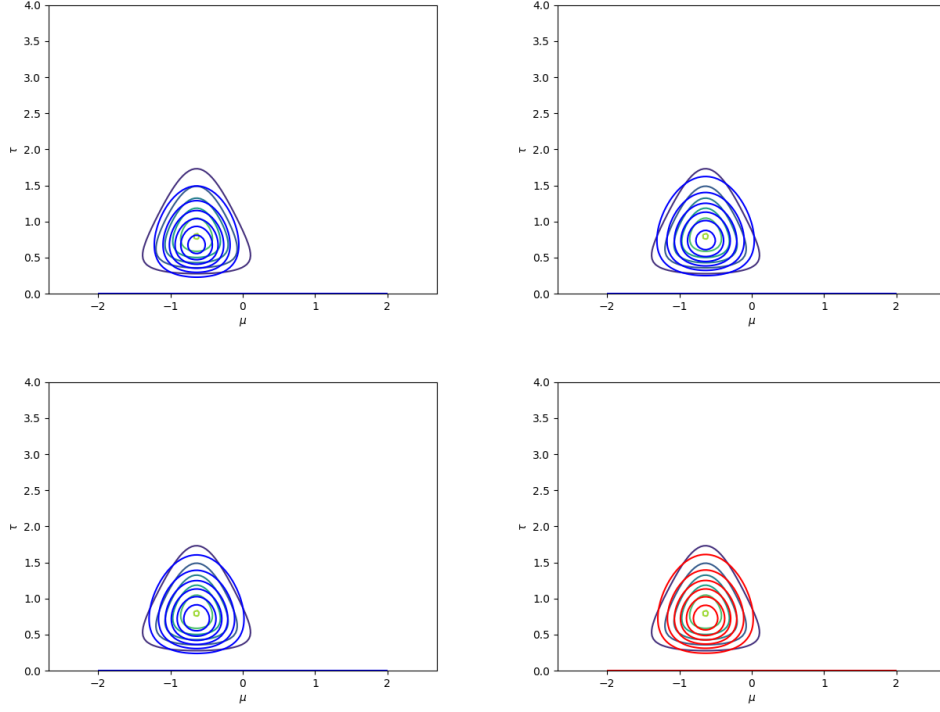


Figure 1: Simple VI with $\mu_0 = \lambda_0 = a_0 = b_0 = 0$, $b_N = 5$, and $\lambda_N = 5$.

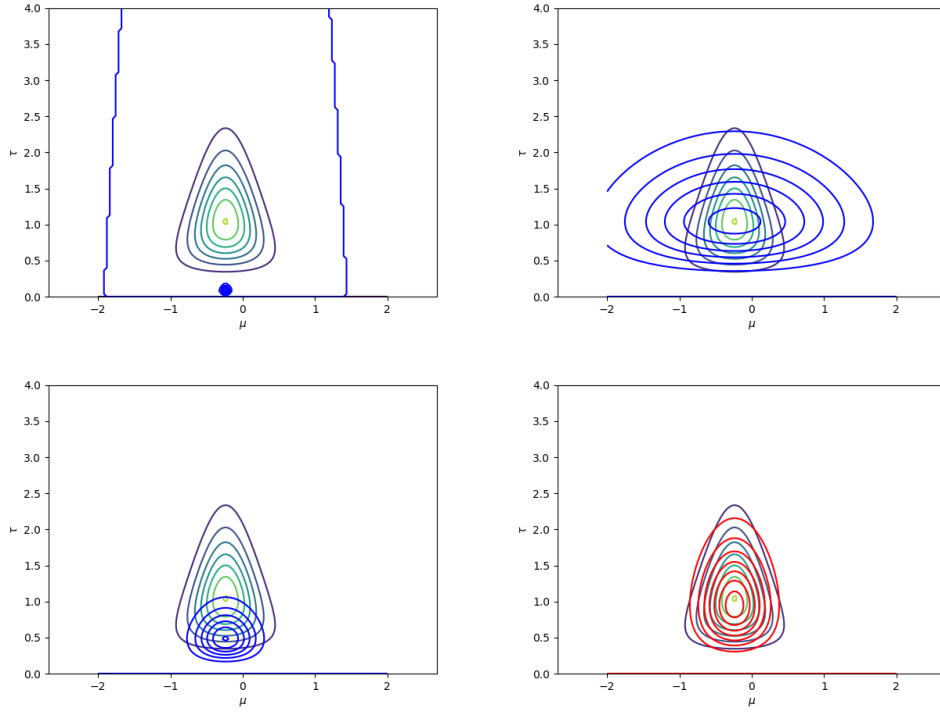


Figure 2: Simple VI with $\mu_0 = \lambda_0 = a_0 = b_0 = 0$, $b_N = 0.1$, and $\lambda_N = 0.1$.

By setting the value of the initial values b_N and λ_N to smaller values which are much different to the true values in the true posterior distribution, the converging procedure is slower and requires more iterations to reach the convergence.

3. In this case, the number of data set is $N = 100$ and the initial values are $b_N = 5$ and $\lambda_N = 5$. The result is illustrated in fig 3.

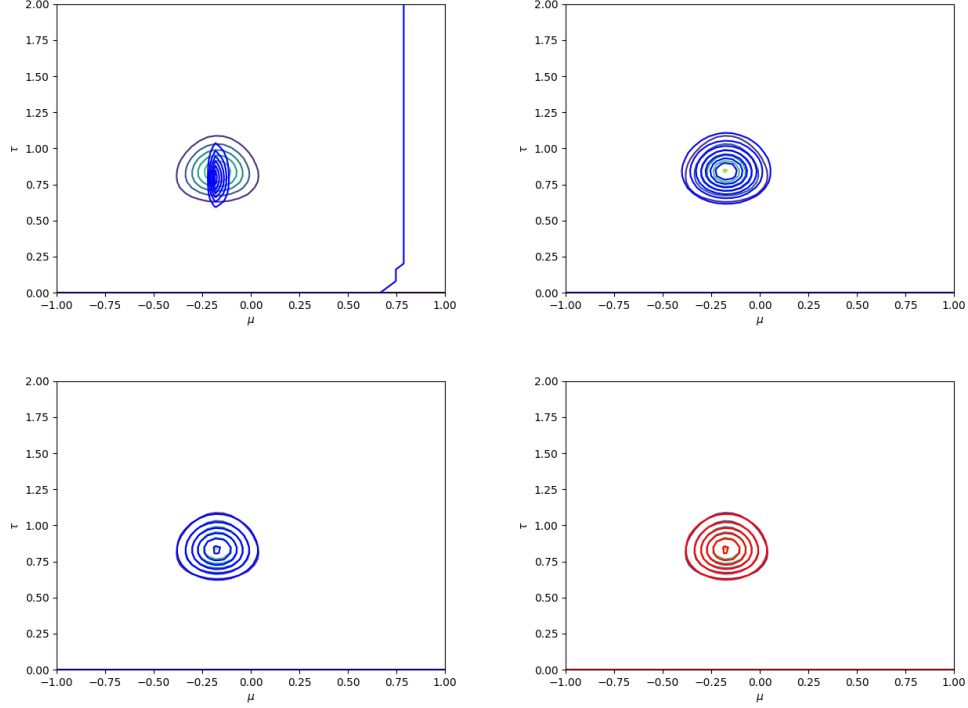


Figure 3: Simple VI with $\mu_0 = \lambda_0 = a_0 = b_0 = 0$, $b_N = 5$, and $\lambda_N = 5$.

In this case, we use 100 data sets generated from a normal distribution. The true posterior distribution for μ and τ looks more like a circular shape. And since we have more data sets, we would be confident to the result and the convergence is quick at the same time.

5 Mixture of trees with observable variables

Question 2.5.15: *Implement this EM algorithm.*

For each n, k the responsibilities is

$$\begin{aligned} r_{n,k} &= \frac{\pi_k p(x^n | T_k, \Theta_k)}{p(x^n)} \\ &= \frac{\pi_k p(x^n | T_k, \Theta_k)}{\sum_{k=1}^K \pi_k p(x^n | T_k, \Theta_k)}. \end{aligned}$$

The algorithm is implemented below, where the function in *Kruskal_v1.py* is slightly modified so that the result of the maximum spanning tree can be returned:


```

from Kruskal_v1 import Graph
2 from Tree import TreeMixture, Tree, Node

import numpy as np
import sys

7 epsilon = sys.float_info.epsilon

def em_algorithm(seed_val, samples, num_clusters, max_num_iter=100):
    print("Running EM algorithm...")

12     # Set threshold for convergence
    THRES = 1e-4

    # Set rounds for sieving
    num_sieving = 10

17     # Get the dimension of the data
    num_samples = np.size(samples, 0)
    num_nodes = np.size(samples, 1)

22     # Sieving
    np.random.seed(seed_val)
    seeds = np.random.randint(0, 100000000, num_sieving)
    last_loglikelihoods = []
    tms = []
27     for seed in seeds:
        np.random.seed(seed)
        tm = TreeMixture(num_clusters=num_clusters, num_nodes=num_nodes)
        tm.simulate_pi(seed_val=seed)
        tm.simulate_trees(seed_val=seed)
32         tm_loglikelihood, tm = em_helper(tm, samples, num_clusters, max_num_iter=10)
        last_loglikelihoods.append(tm_loglikelihood[-1])
        tms.append(tm)

    # Main procedure for EM algorithm
37     print("=> Sieving finished")
    seed = seeds[last_loglikelihoods.index(max(last_loglikelihoods))]
    tm = TreeMixture(num_clusters=num_clusters, num_nodes=num_nodes)
    tm.simulate_pi(seed_val=seed)
    tm.simulate_trees(seed_val=seed)
42     loglikelihood, tm = em_helper(tm, samples, num_clusters, max_num_iter=
max_num_iter)

    print("=> EM finished")
    topology_list = []
    theta_list = []
47     for t in tm.clusters:
        topology_list.append(t.get_topology_array())
        theta_list.append(t.get_theta_array())
        loglikelihood = np.array(loglikelihood)
        topology_list = np.array(topology_list)
52     theta_list = np.array(theta_list)

    return loglikelihood, topology_list, theta_list, tm

def em_helper(tm, samples, num_clusters, max_num_iter=10):
57     num_samples = np.size(samples, 0)

```

```

num_nodes = np.size(samples, 1)

loglikelihood = []
for iter in range(max_num_iter):
62     print("===== "+str(iter)+"-th iteration =====")
)

# Step 1: Compute the responsibilities
r = np.ones((num_samples, num_clusters))

67     for n, x in enumerate(samples):
        for k, t in enumerate(tm.clusters):
            r[n,k] *= tm.pi[k]
            visit_list = [t.root]
            while len(visit_list) is not 0:
                cur_node = visit_list[0]
72                 visit_list = visit_list[1:]
                visit_list = visit_list + cur_node.descendants
                if cur_node.ancestor is None:
                    r[n,k] *= cur_node.cat[x[int(cur_node.name)]]
                else:
77                     r[n,k] *= cur_node.cat[x[int(cur_node.ancestor.name)]] [x[int
(cur_node.name)]]

            r += epsilon
            marginal = np.reshape(np.sum(r, axis=1), (num_samples, 1))
            loglikelihood.append(np.sum(np.log(marginal)))
82            marginal_expand = np.repeat(marginal, num_clusters, axis=1)
            r /= marginal_expand

# Step 2: Update categorical distribution
tm.pi = np.mean(r, axis=0)

87

# Step 3: Construct directed graphs
denom = np.sum(r, axis=0)
q = np.zeros((num_nodes, num_nodes, 2, 2, num_clusters)) # (s, t, a, b, k)
92     for s in range(num_nodes):
        for t in range(num_nodes):
            for a in range(2):
                for b in range(2):
                    index = np.where((samples[:,(s,t)]==[a,b]).all(1))[0]
                    number = np.sum(r[index], axis=0)
97                     q[s, t, a, b] = number / denom

            q += epsilon

            q_s = np.zeros((num_nodes, 2, num_clusters))
            102             for s in range(num_nodes):
                for a in range(2):
                    index = np.where(samples[:, s]==a)
                    number = np.sum(r[index], axis=0)
                    q_s[s,a] = number / denom
            q_s += epsilon

107
            I = np.zeros((num_nodes, num_nodes, num_clusters)) # (s, t, k)
            for s in range(num_nodes):
                for t in range(num_nodes):
                    for a in range(2):
                        for b in range(2):
112                             I[s,t] += q[s,t,a,b] * np.log(q[s,t,a,b] / q_s[s,a] / q_s[t,
b])

```

```

clusters = []
for k in range(num_clusters):
    g = Graph(num_nodes)
    for s in range(num_nodes):
        for t in range(s+1, num_nodes):
            g.addEdge(s, t, I[s, t, k])

# Step 4: Construct maximum spanning trees
edges = np.array(g.maximum_spanning_tree())[ :, 0:2]
topology_array = np.zeros(num_nodes)
topology_array[0] = np.nan
visit_list = [0]
while len(visit_list) != 0:
    cur_node = visit_list[0]
    index = np.where(edges==cur_node)
    index = np.transpose(np.stack(index))
    visit_list = visit_list[1:]
    for id in index:
        child = edges[id[0], 1-id[1]]
        topology_array[int(child)] = cur_node
        visit_list.append(int(child))
    if np.size(index) is not 0:
        edges = np.delete(edges, index[:,0], axis=0)

tree = Tree()
tree.load_tree_from_direct_arrays(topology_array)
tree.k = 2
tree.alpha = [1.0] * 2

# Step 5: Update CPDs
visit_list = [tree.root]
while len(visit_list) != 0:
    cur_node = visit_list[0]
    visit_list = visit_list[1:]
    visit_list = visit_list + cur_node.descendants
    if cur_node.ancestor is None:
        cur_node.cat = q_s[int(cur_node.name) :, k].tolist()
    else:
        cat = q[int(cur_node.ancestor.name), int(cur_node.name) :, :, k]
        cur_node.cat = [cat[0].tolist(), cat[1].tolist()]

    clusters.append(tree)
tm.clusters = clusters

return loglikelihood, tm

```

Listing 2: EM algorithm for mixture of trees with observable variables.

Question 2.5.16: *Apply your algorithm to the provided data and show how well you reconstruct the mixtures. First, compare the real and inferred trees with the unweighted Robinson-Foulds (aka symmetric difference) metric. Do the trees have similar structure (don't worry if the inferred trees don't match with the real trees)? Then, compare the likelihoods of real and inferred mixtures. Finally, simulate more data and analyze the results (try to find some interesting and more challenging cases).*

- **q_2_5_tm_10node_20sample_4clusters:** The result of the EM algorithm is illustrated in

fig 4.

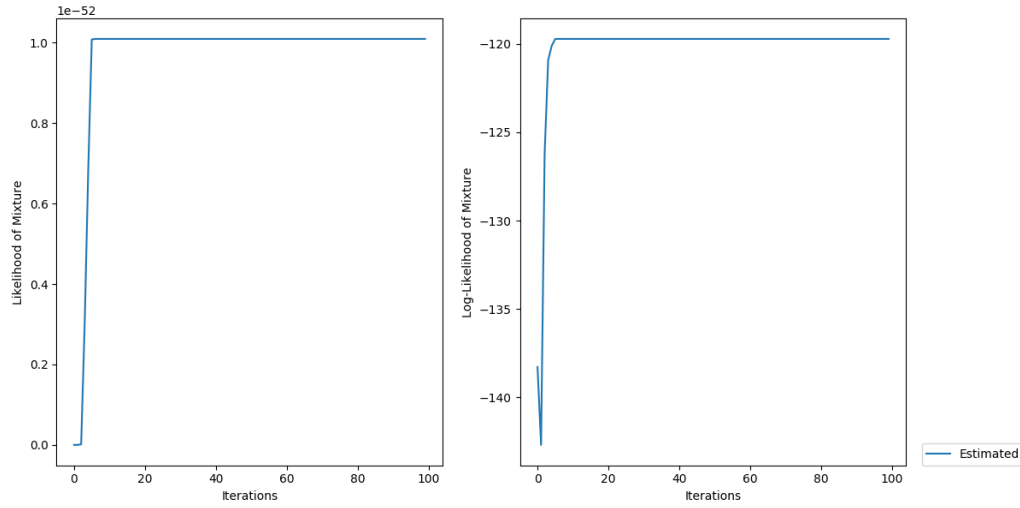


Figure 4: The likelihood and log-likelihood at each iteration of the EM algorithm for data *q_2_5_tm_10node_20sample_4clusters*.

The comparison of the real and inferred trees with unweighted RF metric is show in tab 1.

Table 1: RF distance between the true tree mixture and the result obtained from the EM algorithm for data *q_2_5_tm_10node_20sample_4clusters*, where the rows represents the obtained tree mixture and the columns represents the true tree mixture.

	0	1	2	3
0	9	8	10	10
1	9	12	10	10
2	8	7	9	9
3	8	11	11	9

The log-likelihood of the result tree mixture obtained by the EM algorithm is -119.724756 , while the log-likelihood of the ground truth is -113.143100 .

Finally, by simulating 1000 data points from the true tree mixture, the result is shown as

- *q_2_5_tm_20node_50sample_4clusters*:
- *q_2_5_tm_20node_20sample_4clusters*:

Question 2.5.17: Simulate new tree mixtures with different number of nodes, samples and clusters. Try to find some interesting cases. Analyze your results as in the previous question.

6 Super epicentra - EM

Question 2.6.18: Derive an EM algorithm for the model.

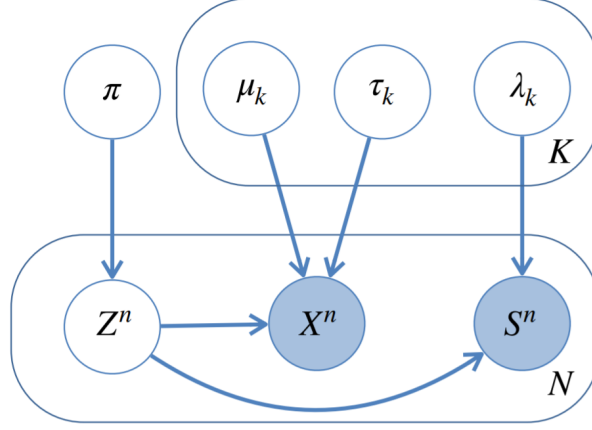


Figure 5: Mixture of components modeling location and strengths of earthquakes associated with a super-epicentra. In the figure, $\mu_k = (\mu_{k,1}, \mu_{k,2})$ and $\tau_k = (\tau_{k,1}, \tau_{k,2})$.

The conditional distribution of X^n, S^n given a particular value for Z^n is

$$\begin{aligned} p(X^n, S^n | Z_k^n = 1) &= \mathcal{N}(X^n | \mu_k, \tau_k) \text{Poisson}(S^n | \lambda_k) \\ &= \frac{\sqrt{\tau_{k,1}\tau_{k,2}}}{2\pi} \exp \left[-\frac{\tau_{k,1}}{2} (X_1^n - \mu_{k,1})^2 - \frac{\tau_{k,2}}{2} (X_2^n - \mu_{k,2})^2 \right] \frac{\lambda_k^{S^n}}{S^n!} \exp[-\lambda_k] \\ &= \frac{\sqrt{\tau_{k,1}\tau_{k,2}}}{2\pi} \frac{\lambda_k^{S^n}}{S^n!} \exp \left[-\frac{\tau_{k,1}}{2} (X_1^n - \mu_{k,1})^2 - \frac{\tau_{k,2}}{2} (X_2^n - \mu_{k,2})^2 - \lambda_k \right] \end{aligned}$$

which can also be written in the form

$$p(X^n, S^n | Z^n) = \prod_{k=1}^K p(X^n, S^n | Z_k^n = 1)^{Z_k^n}.$$

The joint distribution is given by

$$p(X^n, S^n, Z^n) = p(Z^n) p(X^n, S^n | Z^n),$$

and the marginal distribution of X^n, S^n is then obtained by summing the joint distribution over all possible states of Z^n to give

$$p(X^n, S^n) = \sum_{k=1}^K \pi_k p(X^n, S^n | Z_k^n = 1).$$

Another quantity that will play an important role is the conditional probability of Z^n given X^n, S^n . We shall use $\gamma(Z_k^n)$ to denote $p(Z_k^n = 1 | X^n, S^n)$, whose value can be found using Bayes' theorem

$$\begin{aligned} \gamma(Z_k^n) \equiv p(Z_k^n = 1 | X^n, S^n) &= \frac{p(Z_k^n = 1) p(X^n, S^n | Z_k^n = 1)}{\sum_{j=1}^K p(Z_j^n = 1) p(X^n, S^n | Z_j^n = 1)} \\ &= \frac{\pi_k p(X^n, S^n | Z_k^n = 1)}{\sum_{j=1}^K \pi_j p(X^n, S^n | Z_j^n = 1)}. \end{aligned}$$

The M-step in general EM algorithm is to evaluate θ^{new} given by

$$\theta^{\text{new}} = \arg_{\theta} \max \mathcal{Q}(\theta, \theta^{\text{old}})$$

where

$$\begin{aligned} \mathcal{Q}(\theta, \theta^{\text{old}}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{S}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{S}, \mathbf{Z}|\theta) \\ &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{S}, \theta^{\text{old}}) \ln p(\mathbf{Z}|\theta) p(\mathbf{X}, \mathbf{S}|\mathbf{Z}, \theta) \end{aligned}$$

Setting the derivatives of with respect to (only for the cases that $Z^n = 1$). ?????

EM algorithm:

1. Initialize the means μ_k , precision τ_k , Poisson parameter λ_k and mixing coefficients π_k , and evaluate the initial value of the log likelihood.
2. **E Step.** Evaluate the responsibilities using the current parameter values

$$\gamma(Z_k^n) = \frac{\pi_k p(X^n, S^n | Z_k^n = 1)}{\sum_{j=1}^K \pi_j p(X^n, S^n | Z_j^n = 1)}.$$

3. **M Step.** Re-estimate the parameters using the current responsibilities

$$\begin{aligned} \mu_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(Z_k^n) X^n \\ \tau_{k,1}^{\text{new}} &= \frac{N_k}{\sum_{n=1}^N \gamma(Z_k^n) (X_1^n - \mu_{k,1}^{\text{new}})^2} \\ \tau_{k,2}^{\text{new}} &= \frac{N_k}{\sum_{n=1}^N \gamma(Z_k^n) (X_2^n - \mu_{k,2}^{\text{new}})^2} \\ \lambda_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(Z_k^n) S^n \\ \pi_k^{\text{new}} &= \frac{N_k}{N} \end{aligned}$$

where

$$N_k = \sum_{n=1}^N \gamma(Z_k^n).$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}, \mathbf{S} | \boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\lambda}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(X^n | \mu_k, \tau_k) \text{Poisson}(S^n | \lambda_k) \right\}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Question 2.6.19: *Implement your EM algorithm.*

```

def _do_estep(self, X, S):
    """
    E-step
    """
    r = np.zeros((self.n_row, self.n_components)) # r[n,k]
    for n in range(self.n_row):
        for k in range(self.n_components):
            r[n,k] = self.weights[k] \
                * multivariate_normal(self.means[k], self.covs[k]).pdf(X[n]) \
                * poisson(self.rates[k]).pmf(S[n])

    r += epsilon
    marginal = np.sum(r, axis=1).reshape((self.n_row, 1))
    marginal = np.repeat(marginal, self.n_components, axis=1)
    r /= marginal

    self.r = r
    return self

def _do_mstep(self, X, S):
    """M-step, update parameters"""
    N = np.sum(self.r, axis=0) # N[k] = Nk
    N_expand = np.repeat(N.reshape((self.n_components, 1)), self.n_col, axis=1)
    self.means = np.dot(np.transpose(self.r), X) / N_expand

    for k in range(self.n_components):
        mean_expand = np.repeat(np.atleast_2d(self.means[k]), self.n_row, axis=0)
        r_expand = np.repeat(self.r[:,k].reshape((self.n_row, 1)), self.n_col, axis=1)
        variance = np.sum(r_expand * (X - mean_expand)**2, axis=0) / N[k]
        self.covs[k] = np.diag(variance)

    self.rates = np.dot(S, self.r) / N

    self.weights = N / self.n_row

    return self

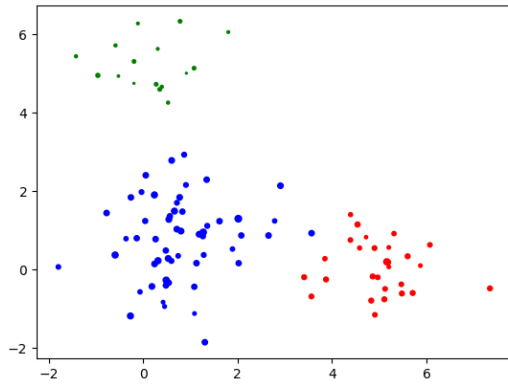
def _compute_log_likelihood(self, X, S):
    """compute the log likelihood of the current parameter"""
    log_likelihood = 0
    for n in range(self.n_col):
        likelihood = 1
        for k in range(self.n_components):
            likelihood *= self.weights[k] \
                * multivariate_normal(self.means[k], self.covs[k]).pdf(X[n]) \
                * poisson(self.rates[k]).pmf(S[n])
        log_likelihood += np.log(likelihood)

    return log_likelihood

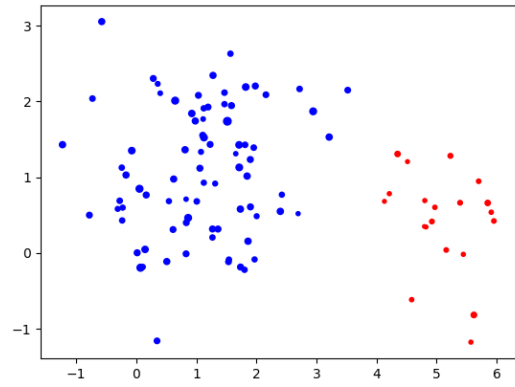
```

Listing 3: EM algorithm for super epicentra.

Question 2.6.20: *Apply it to the data provided separately, give an account of the success, and provide visualizations for a couple of examples.*



(a) Data set 1.



(b) Data set 2.

Figure 6: Ground truth for data sets

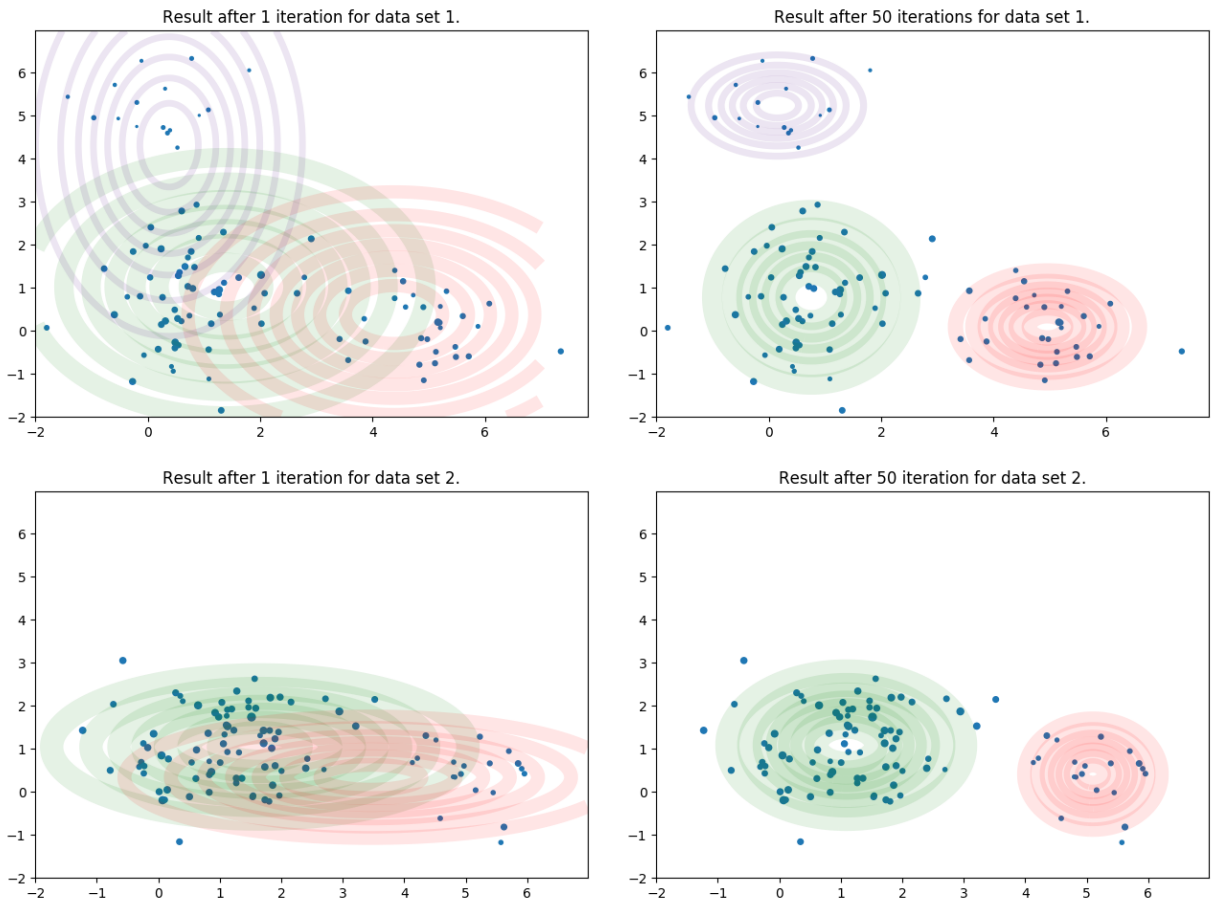


Figure 7: Result of the EM algorithm for different data sets.

7 Super epicentra - VI

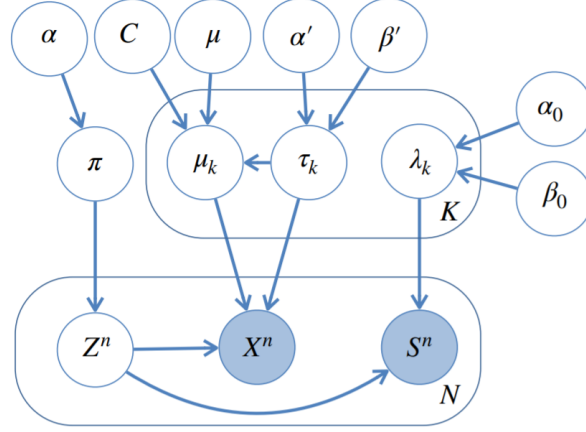


Figure 8: The K super epicentra model with priors.

Question 2.7.21: *Derive a VI algorithm that estimates the posterior distribution for this model.*

The VI algorithm is derived based on [1].
Since $\pi \sim \text{Dir}(\alpha)$, we have

$$\begin{aligned} p(\pi|\alpha) &= p(\pi_1, \dots, \pi_K | \alpha_1, \dots, \alpha_K) \\ &= \frac{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \pi_k^{\alpha_k-1} \\ &= \frac{1}{B(\alpha)} \prod_{k=1}^K \pi_k^{\alpha_k-1}. \end{aligned}$$

Also, for other latent variables, we have

$$\begin{aligned} p(\mu_k|\tau_k) &= \mathcal{N}(\mu_k | \mu, (C\tau_k)^{-1}) \\ p(\tau_k) &= \text{Gam}(\tau_k | \alpha', \beta') \\ p(\lambda_k) &= \text{Gam}(\lambda_k | \alpha_0, \beta_0), \end{aligned}$$

and from the previous question, we know that

$$\begin{aligned} p(X^n, S^n | Z_k^n = 1, \mu_k, \tau_k, \lambda_k) &= \mathcal{N}(X^n | \mu_k, \tau_k) \text{Poisson}(S^n | \lambda_k) \\ &= \frac{\sqrt{\tau_{k,1}\tau_{k,2}}}{2\pi} \exp\left[-\frac{\tau_{k,1}}{2}(X_1^n - \mu_{k,1})^2 - \frac{\tau_{k,2}}{2}(X_2^n - \mu_{k,2})^2\right] \frac{\lambda_k^{S^n}}{S^n!} \exp[-\lambda_k] \\ &= \frac{\sqrt{\tau_{k,1}\tau_{k,2}}}{2\pi} \frac{\lambda_k^{S^n}}{S^n!} \exp\left[-\frac{\tau_{k,1}}{2}(X_1^n - \mu_{k,1})^2 - \frac{\tau_{k,2}}{2}(X_2^n - \mu_{k,2})^2 - \lambda_k\right]. \end{aligned}$$

The joint distribution of all of the random variables is given by

$$p(\mathbf{X}, \mathbf{S}, \mathbf{Z}, \pi, \mu, \tau, \lambda) = p(\mathbf{X}|\mathbf{Z}, \mu, \tau)p(\mathbf{S}|\mathbf{Z}, \lambda)p(\mathbf{Z}|\pi)p(\pi)p(\mu|\tau)p(\tau)p(\lambda).$$

A variational distribution which factorizes between the latent variables and the parameters is given by

$$q(\mathbf{Z}, \pi, \mu, \tau, \lambda) = q(\mathbf{Z})q(\pi, \mu, \tau, \lambda).$$

The log of the optimized factor is then given by

$$\begin{aligned} \ln q^*(\mathbf{Z}) &= \mathbb{E}_{\pi, \mu, \tau, \lambda} [\ln p(\mathbf{X}, \mathbf{S}, \mathbf{Z}, \pi, \mu, \tau, \lambda)] + \text{const} \\ &= \mathbb{E}_{\pi} [\ln p(\mathbf{Z}|\pi)] + \mathbb{E}_{\mu, \tau} [\ln p(\mathbf{X}|\mathbf{Z}, \mu, \tau)] + \mathbb{E}_{\lambda} [\ln p(\mathbf{S}|\mathbf{Z}, \lambda)] + \text{const} \\ &= \sum_{n=1}^N \sum_{k=1}^K Z_k^n \ln \rho_{n,k} + \text{const} \end{aligned}$$

where we have defined

$$\begin{aligned} \ln \rho_{n,k} &= \mathbb{E} [\ln \pi_k] + \frac{1}{2} \mathbb{E} [\ln |\tau_k|] - \ln(2\pi) \\ &\quad - \frac{1}{2} \mathbb{E}_{\mu_k, \tau_k} [(X^n - \mu_k)^T \tau_k (X^n - \mu_k)] \\ &\quad + S^n \mathbb{E} [\ln \lambda_k] - \mathbb{E} [\lambda_k] - \ln(S^n!). \end{aligned}$$

Taking the exponential of both sides of the log of the optimized factor and normalize it, we obtain

$$q^*(\mathbf{Z}) = \prod_{n=1}^N \prod_{k=1}^K r_{nk}^{Z_k^n}$$

where

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^K \rho_{nj}}.$$

For the discrete distribution $q^*(\mathbf{Z})$ we have the standard result

$$\mathbb{E}[Z_k^n] = r_{nk}$$

from which we see that the quantities r_{nk} are playing the role of responsibilities. Define

$$N_k = \sum_{n=1}^N r_{nk}.$$

Now let us consider the factor $q(\pi, \mu, \tau, \lambda)$ in the variational posterior distribution, we have

$$\begin{aligned} \ln q^*(\pi, \mu, \tau, \lambda) &= \ln p(\pi) + \mathbb{E}_{\mathbf{Z}} [\ln p(\mathbf{Z}|\pi)] + \text{const} \\ &\quad + \sum_{k=1}^K \ln p(\mu_k, \tau_k) + \sum_{k=1}^K \sum_{n=1}^N \mathbb{E}[Z_k^n] \ln \mathcal{N}(X^n | \mu_k, \tau_k^{-1}) \\ &\quad + \sum_{k=1}^K \ln p(\lambda_k) + \sum_{k=1}^K \sum_{n=1}^N \mathbb{E}[Z_k^n] \ln \text{Poisson}(S^n | \lambda_k). \end{aligned}$$

Since the right-hand side of this expression decomposes into a sum of terms involving only π together with terms only involving μ and τ and terms only involving λ , which implies that the

variational posterior $q(\pi, \mu, \tau, \lambda)$ factorizes to give $q(\pi)q(\mu, \tau)q(\lambda)$, and by summing over k of terms involving μ_k and τ_k , and λ_k leads to the further factorization

$$q(\pi, \mu, \tau, \lambda) = q(\pi) \prod_{k=1}^K q(\mu_k, \tau_k) q(\lambda_k).$$

- Identifying the terms that depend on π , we have

$$\ln q^*(\pi) = (\alpha - 1) \sum_{k=1}^K \ln \pi_k + \sum_{k=1}^K \sum_n^N r_{nk} \ln \pi_k + \text{const}$$

and taking the exponential of both sides, we have

$$q^*(\pi) = \text{Dir}(\pi | \alpha^*)$$

where α^* has components α_k^* given by

$$\alpha_k^* = \alpha_k + N_k.$$

- Similarly, by identifying the terms that depends on μ_k and τ_k , we have

$$\begin{aligned} \ln q^*(\mu_k, \tau_k) &= \sum_{n=1}^N r_{nk} \left(\frac{1}{2} \ln \tau_k - \frac{1}{2} \tau_k (X^n - \mu_k)^2 \right) \\ &\quad + \frac{1}{2} \ln \tau_k - \frac{1}{2} C (\mu_k - \mu)^2 \\ &\quad + (\alpha' - 1) \ln \tau_k - \beta' \tau_k. \end{aligned}$$

So, we have

$$\begin{aligned} q^*(\mu_k | \tau_k) &= \mathcal{N}(\mu_k | \mu^*, \tau^*) \\ \tau^* &= (C + N_k) \tau_k \\ \mu^* &= \frac{\left(\tau_k \sum_{n=1}^N r_{nk} X^n \right) + C \tau_k \mu}{\tau^*}, \end{aligned}$$

and

$$\begin{aligned} q^*(\tau_k) &= \text{Gam}(\tau_k | \alpha'^*, \beta'^*) \\ \alpha'^* &= \alpha' + N_k \\ \beta'^* &= \beta' + \frac{1}{2} C \mu^2 + \frac{1}{2} \sum_{n=1}^N r_{nk} X^{n2}. \end{aligned}$$

- Finally, by identifying the terms that depends on λ_k , we have

$$\begin{aligned} \ln q^*(\lambda_k) &= (\alpha_0 - 1) \ln \lambda_k - \beta_0 \lambda_k + \text{const} \\ &\quad + \sum_{n=1}^N (r_{nk} (S^n \ln \lambda_k - \lambda_k - \ln(S^n!))). \end{aligned}$$

So, we have

$$\begin{aligned}q^*(\lambda_k) &= \text{Gam}(\lambda_k | \alpha_0^*, \beta_0^*) \\ \alpha_0^* &= \alpha_0 + \sum_{n=1}^N r_{nk} S^n \\ \beta_0^* &= \beta + N_k.\end{aligned}$$

8 Sampling from a tree GM

Question 2.8.22: *Derive these algorithms.*

Question 2.8.23: *Implement your bottom up DP algorithm for the probability of generating an odd sum output.*

Question 2.8.24: *Implement your sampling algorithm.*

Question 2.8.25: *Apply your algorithm to the graphical model and data provided separately.*

9 Failing components VI

Question 2.9.26: *Derive a VI algorithm that estimates the posterior distribution for this model.*

References

- [1] C. M. Bishop, *Pattern recognition and machine learning*. Information science and statistics, New York: Springer, 2006.
- [2] J. Kruschke, *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014.